Лабораторная работа 8.

Черновик 0.7

Целью работы является знакомство студентов с двумерными динамическими массивами.

Студенты должны получить и закрепить на практике следующие знания и умения:

1. Выделение и освобождение памяти под двумерные динамические массивы.
2. Обработка матриц и текстовых файлов.
3. Организация корректной работы с ресурсами (динамически выделенная память, файловые описатели).
4. Использование в программе аргументов командной строки.
5. Контроль правильности работы с динамической памятью с помощью специального ПО.

1. Общее задание

1. Исходный код лабораторной работы располагается в отдельной ветке lab_08. В ветке lab_08 создается папка lab_08_X_YZ, где вместо X указывается номер индивидуального задания, а вместо Y и Z – форматы входного и выходного файлов, закодированные числом (см. ниже). Варианты распределяются преподавателем.
2. Исходный код должен соответствовать правилам оформления исходного кода.
3. Решение задачи оформляется как многофайловый проект. Для сборки проекта используется утилита make. В сценарии сборки должна присутствовать цель app.exe, с помощью которой собирается приложение, и цель test.exe для сборки тестов.
4. Для задачи студентом подготавливаются тестовые данные, которые демонстрируют правильность ее работы (функциональные тесты). Входные данные должны располагаться в файлах in_z.txt (возможно таких файлов будет несколько), выходные out_z.txt, где z – номер тестового случая. Тестовые данные готовятся и помещаются под версионный контроль еще до того, как появится реализация задачи.

2. Индивидуальное задание

Написать программу для работы с матрицами, которая реализует сложение матриц, умножение матриц и указанную ниже операцию.

Память под матрицы выделяется динамически.

Исходные матрицы читаются из файла, результирующие матрицы записываются в файл. Форматы входных и выходных файлов могут различаться. Один файл содержит одну матрицу.

Тестирование выполняется с помощью сравнения полученного результата с ожидаемым. При этом нужно помнить, что сравнивать вещественные числа на равенство можно только с заданной точностью.

Имена файлов и выполняемая операция указывается через параметры командной строки.
Формат запуска приложения должен быть следующим:

```
app.exe action mtr_1.txt [mtr_2.txt] res.txt
```

Возможные значения action:

- **a** сложение;
- **m** умножение;
- **o** операция по варианту (для нее mtr_2.txt не указывается);
- **h** выдача справочной информации.

Если операция по варианту «решение СЛАУ», то столбец свободных членов дописывается к матрице коэффициентов и матрица размером *n x (n+1)* помещается в файл *mtr_1.txt*. Столбец решений сохраняется в файле *res.txt* в виде матрицы (т.е. на первой строчке указывается размер n 1, а со следующей строчки идут значения элементов).

Если операция по варианту «вычисление определителя», то файл *res.txt* содержит число – значение определителя.

2.1. Способы выделения памяти

1. Массив указателей на строки.
2. Объединенный подход, способ 1.
3. Объединенный подход, способ 2.

Замечание.
Крайне желательно продумать такую реализацию функций выделения памяти под матрицу и освобождения памяти из-под матрицы, чтобы можно было легко «переключаться» между разными способами выделения памяти.

2.2. Форматы файлов

1. Простой формат.
2. Координатный формат.

Простой формат
Количество строк и столбцов матрицы указывается в первой строке файла, остальные строки содержат сами элементы.

```
2 3
0 1 2
3 4 5
```

Координатный формат
Количество строк и столбцов матрицы и количество ненулевых элементов указывается на первой строке матрицы. Остальные строки содержат тройки чисел: «номер строки» «номер столбца» «значение элемента». Индексация выполняется с единицы. Указываются значения только тех элементов, которые отличны от нуля.

```
2 3 5
1 2 1
1 3 2
2 1 3
2 2 4
2 3 5
```

2.3. Операция, реализуемая по варианту

1. Метод Гаусса с выбором главного (ведущего) элемента по столбцу.
2. Метод Гаусса с выбором главного (ведущего) элемента по строке.
3. Метод Гаусса с выбором главного (ведущего) элемента по активной подматрице.
4. Вычисление определителя с помощью разложения по строке.
5. Вычисление определителя с помощью разложения по столбцу.
6. Вычисление обратной матрицы методом Гаусса.
7. Вычисление обратной матрицы методом элементарных преобразований.
8. Вычисление определителя методом Гаусса.

Пояснения к 6.
Обратная матрица ищется через решение системы Ax=f с различными правыми частями. Правая часть последовательно пробегает значения столбцов e(j) единичной матрицы E, при этом для каждой из них найденное решение системы Ax=f образует j-ый столбец искомой обратной матрицы.

Пояснения к 7.
К исходной матрице справа приписывается единичная матрица того же порядка: (A|E). С помощью элементарных преобразований строк и столбцов левая "половина" приводится к единичной, совершая одновременно точно такие же преобразования над правой матрицей.

Замечание 1.
На практике метод Гаусса не используется без выбора главного элемента. Поэтому если не оговорено специально и в формулировке задания указан метод Гаусса, считайте, что это метод Гаусса с выбором главного элемента по столбцу.

Замечание 2.
Для сравнения вещественных чисел можно использовать не только абсолютную,

```
if (fabs(a - b) <= eps)
{
    // Числа равны с абсолютной точностью eps (0 < eps < 1)
}
```

но и относительную точность:

```
if (fabs(a - b) <= eps * fmax(fabs(a), fabs(b))
{
    // Числа равны с относительной точностью eps (0 < eps < 1)
}
```

2.4. Примеры именования папок

Вариант операции	Входной формат	Выходной формат	Имя папки
2	1	1	lab_08_2_11

5	2	1	lab_08_5_21
3	1	2	lab_08_3_12

Способ выделения памяти под матрицу не участвует в формировании имени папки!